

BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL

BBBBBBBB	AAAAAA	SSSSSSSS	IIIIII	NN	NN	IIIIII	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF
BBBBBBBB	AAAAAA	SSSSSSSS	IIIIII	NN	NN	IIIIII	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF
BB	BB	AA	II	NN	NN	II	DD	DD	EE
BB	BB	AA	II	NN	NN	II	DD	DD	EE
BB	BB	AA	II	NNNN	NN	II	DD	DD	EE
BB	BB	AA	II	NNNN	NN	II	DD	DD	EE
BBBBBBBB	AA	SSSSSS	II	NN	NN	II	DD	DD	EEEEEEEE
BBBBBBBB	AA	SSSSSS	II	NN	NN	II	DD	DD	EEEEEEEE
BB	AAAAAAAAA	SS	II	NN	NNNN	II	DD	DD	EE
BB	AAAAAAAAA	SS	II	NN	NNNN	II	DD	DD	EE
BB	AA	SS	II	NN	NN	II	DD	DD	EE
BB	AA	SS	II	NN	NN	II	DD	DD	EE
BBBBBBBB	AA	SSSSSSSS	IIIIII	NN	NN	IIIIII	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF
BBBBBBBB	AA	SSSSSSSS	IIIIII	NN	NN	IIIIII	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF

....
....
....
....

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS


```

1 0001 0 MODULE BASSINIT_DEF (
2 0002 0   IDENT = '1-012'
3 0003 0   ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 *   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 *   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 *   ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 *   TRANSFERRED.
18 0018 1 *
19 0019 1 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 *   CORPORATION.
22 0022 1 *
23 0023 1 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY: BASIC-PLUS-2 Frame Support
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1   These routines set up and tear down frames for BASIC-PLUS-2.
37 0037 1   Frames are used for main routines, external functions,
38 0038 1   external subroutines, internal functions (both DEFs and DEF*s)
39 0039 1   internal subroutines (GOSUBs) and condition handlers.
40 0040 1
41 0041 1 ENVIRONMENT: VAX-11 user mode
42 0042 1
43 0043 1 AUTHOR: John Sauter, CREATION DATE: 10-Oct-78
44 0044 1
45 0045 1 MODIFIED BY:
46 0046 1
47 0047 1 1-001 - Original. This is just a shell.
48 0048 1 1-002 - Copy code from BASINIT. JBS 07-FEB-1979
49 0049 1 1-003 - Remove tests for scale factors and long/double, since the
50 0050 1   compiler will not permit a DEF to have different options
51 0051 1   than its containing major procedure. JBS 07-FEB-1979
52 0052 1 1-004 - Change from BASS$ to BSF$ for BASIC stack frame. JBS 08-FEB-1979
53 0053 1 1-005 - Remove BASSB_IN_L_FCD. JBS 09-FEB-1979
54 0054 1 1-006 - When nulling the first string parameter, hold its address
55 0055 1   in an auxiliary variable to avoid a BLISS bug. JBS 12-FEB-1979
56 0056 1 1-007 - Allocate fixed string templates on the stack. JBS 20-MAR-1979
57 0057 1 1-008 - Don't imply that R11 points to a frame. JBS 08-MAY-1979

```


BASS\$INIT_DEF
1-012

H 1
16-Sep-1984 00:35:27
14-Sep-1984 11:55:07

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASINIDEF.B32;1

Page 2
(1)

:	58	0058	1	:	1-009 - Change LIB\$\$ and OT\$\$ to STR\$. JBS 21-MAY-1979
:	59	0059	1	:	1-010 - Use shift operator instead of divide. JBS 11-JUN-1979
:	60	0060	1	:	1-011 - Check for a proper argument list. JBS 03-AUG-1979
:	61	0061	1	:	1-012 - Remove BASSK_WROMATPAC, not used. JBS 19-SEP-1979
:	62	0062	1	:	--
:	63	0063	1	:	
:	64	0064	1	:	!<BLF/PAGE>


```

: 123      0501 1      BASSHANDLER;                      ! handles signals
: 124      0502 1
: 125      0503 1
: 126      0504 1  ! The following are the error codes used in this module.
: 127      0505 1
: 128      0506 1
: 129      0507 1 EXTERNAL LITERAL
: 130      0508 1      BASSK_TOOFEWARG : UNSIGNED (8),      ! Too few arguments
: 131      0509 1      BASSK_TOOMANARG : UNSIGNED (8),      ! Too many arguments
: 132      0510 1      BASSK_SCAFACINT : UNSIGNED (8),      ! Scale factor interlock
: 133      0511 1      BASSK_PROLOSSOR : UNSIGNED (8),      ! Program lost, sorry
: 134      0512 1      BASSK_ARGDONMAT : UNSIGNED (8),      ! Arguments don't match
: 135      0513 1      BASSK_NOTIMP : UNSIGNED (8);          ! Not implemented
: 136      0514 1

```



```
138 0515 1 GLOBAL ROUTINE BASS$INIT_DEF_R8 (
139 0516 1     ARGLIST,
140 0517 1     DATA_RELOC
141 0518 1 ) : NOVALUE BASS$INIT_LINK =
142 0519 1
143 0520 1 ++
144 0521 1 FUNCTIONAL DESCRIPTION:
145 0522 1
146 0523 1     Set up a frame for a BASIC-PLUS-2 DEF. The frame is allocated
147 0524 1     on the stack, and R10 and R9 are set up to point to it.
148 0525 1     The argument tells how to do the allocation.
149 0526 1
150 0527 1 FORMAL PARAMETERS:
151 0528 1
152 0529 1     ARGLIST.ra.v List of information needed to set up the
153 0530 1     frame. See BASIC-PLUS-2/VAX Description
154 0531 1     of Generated Code for details.
155 0532 1     DATA_RELOC.ra.v Address of the major procedure's contribution
156 0533 1     to the data PSECT. This is needed so that the
157 0534 1     argument list can be PIC.
158 0535 1
159 0536 1 IMPLICIT INPUTS:
160 0537 1
161 0538 1     NONE
162 0539 1
163 0540 1 IMPLICIT OUTPUTS:
164 0541 1
165 0542 1     The values of R10 and R9, which point to the automatic
166 0543 1     storage and the temporary storage, respectively.
167 0544 1
168 0545 1 ROUTINE VALUE:
169 0546 1
170 0547 1     NONE
171 0548 1
172 0549 1 COMPLETION CODES:
173 0550 1
174 0551 1     NONE
175 0552 1
176 0553 1 SIDE EFFECTS:
177 0554 1
178 0555 1     Leaves lots of things on the stack for use by the compiled
179 0556 1     BASIC-PLUS-2 code. These things will be removed by
180 0557 1     BAS$END_DEF_R8.
181 0558 1
182 0559 1 --
183 0560 1
184 0561 2 BEGIN
185 0562 2
186 0563 2 EXTERNAL REGISTER
187 0564 2     BSF$A_MAJOR_STG : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME),
188 0565 2     BSF$A_MINOR_STG : REF BLOCK [0, BYTE] FIELD (BSF$MINOR_FRAME),
189 0566 2     BSF$A_TEMP_STG;
190 0567 2
191 0568 2 BUILTIN
192 0569 2     AP,
193 0570 2     FP,
194 0571 2     SP;
```



```
195 0572 2
196 0573 2 MAP
197 0574 2   ARGLIST : REF BLOCK [0, BYTE] FIELD (BASSINIT_ARGS), ! arg list
198 0575 2   AP : REF VECTOR; ! caller's arg list
199 0576 2
200 0577 2 !+
201 0578 2 Define local variables as registers. We cannot have any stack locals
202 0579 2 since we manipulate the stack pointer in this routine.
203 0580 2 -
204 0581 2
205 0582 2 REGISTER
206 0583 2   RETURN_ADDRESS, ! address to return to
207 0584 2   FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD), ! pointer to FCD
208 0585 2   ARRAY_DESC : REF BLOCK [0, BYTE], ! pointer to build array descriptors
209 0586 2   ARRAY_INDEX; ! index for array modification
210 0587 2
211 0588 2 !+
212 0589 2 Save return address because we are going to fool with the stack
213 0590 2 -
214 0591 2   RETURN_ADDRESS = ..SP;
215 0592 2 !+
216 0593 2 Make sure we are passed an argument list we understand.
217 0594 2 -
218 0595 2
219 0596 2   IF (.ARGLIST [BASS$B_IN_V_FCD] NEQ BASS$K_IN_V_FCD) THEN BASS$$SIGNAL (BASS$K_NOTIMP);
220 0597 2
221 0598 2 !+
222 0599 2 Allocate frame control data.
223 0600 2 -
224 0601 2   FMP = .FP;
225 0602 2   SP = .FMP - BSF$K_LENFCDDDEF + %UPVAL;
226 0603 2 !+
227 0604 2 Allocate BSF$A_USER_HAND.
228 0605 2 It is initialized to 0 normally (ON ERROR GOTO 0), but if the
229 0606 2 first statement in the program is ON ERROR GOTO <line number>
230 0607 2 or ON ERROR GO BACK, it is initialized to 1 (ON ERROR GO BACK)
231 0608 2 to prevent a "window" in which error handling is ON ERROR GOTO 0
232 0609 2 no matter what the user wants.
233 0610 2 -
234 0611 2
235 0612 2   IF (((.ARGLIST [BASS$W_IN_FLAGS]) AND (BSF$M_FCD_OEGO)) NEQ 0)
236 0613 2   THEN
237 0614 2     BEGIN
238 0615 2       SP = .SP - %UPVAL;
239 0616 2       .SP = 1;
240 0617 2     END
241 0618 2   ELSE
242 0619 2     BEGIN
243 0620 2       SP = .SP - %UPVAL;
244 0621 2       .SP = 0;
245 0622 2     END;
246 0623 2
247 0624 2 !+
248 0625 2 LOAD Rn (R10)
249 0626 2 -
250 0627 2   BSF$A_MINOR_STG = .SP - 127;
251 0628 2 !+
```



```
252 0629 2 | Initialize parts of the frame control data.
253 0630 2 | -
254 0631 2 | FMP [BSF$A_MARK] = 0;
255 0632 2 | FMP [BSF$A_BASE_R11] = .BSF$A_MAJOR_STG;
256 0633 2 | FMP [BSF$A_BASE_R10] = .BSF$A_MINOR_STG;
257 0634 2 | FMP [BSF$B_LEN_FCD] = BSF$K_LENFCDDDEF;
258 0635 2 | FMP [BSF$B_PROC_CODE] = .ARGLIST [BASS$B_IN_PROC_C];
259 0636 2 | FMP [BSF$W_FCD_FLAGS] = .ARGLIST [BASS$W_IN_FLAGS];
260 0637 2 | FMP [BSF$A_PROC_ID] = .ARGLIST [BASS$L_IN_PROC_I] + .DATA_RELOC;
261 0638 2 | FMP [BSF$A_INIT_ARG] = .ARGLIST;
262 0639 2 | FMP [BSF$L_INIT_REL] = .DATA_RELOC;
263 0640 2 | +
264 0641 2 | Allocate numeric scalars. They are all initialized to zero.
265 0642 2 | -
266 0643 2 |
267 0644 2 | INCR COUNTER FROM 1 TO .ARGLIST [BASS$L_IN_LEN_SC] DO
268 0645 2 | BEGIN
269 0646 2 | SP = .SP - %UPVAL;
270 0647 2 | .SP = 0;
271 0648 2 | END;
272 0649 2 | +
273 0650 2 | Copy formals.
274 0651 2 | -
275 0652 2 |
276 0653 2 |
277 0654 2 | DECR COUNTER FROM MIN (.ARGLIST [BASS$B_IN_NO_FML], ((.AP [0]) AND 255)) TO 1 DO
278 0655 2 | BEGIN
279 0656 2 | SP = .SP - %UPVAL;
280 0657 2 | .SP = .AP [.COUNTER];
281 0658 2 | END;
282 0659 2 | +
283 0660 2 | Allocate and initialize descriptors.
284 0661 2 | -
285 0662 2 | SP = .SP - .ARGLIST [BASS$L_IN_LEN_DT];
286 0663 2 | +
287 0664 2 | Set ARRAY_DESC to point to the space allocated.
288 0665 2 | -
289 0666 2 | ARRAY_DESC = .SP;
290 0667 2 | +
291 0668 2 | Load the space from the template and then modify it based
292 0669 2 | on the modification table.
293 0670 2 | -
294 0671 2 |
295 0672 2 |
296 0673 2 | INCR COUNTER FROM 0 TO ((.ARGLIST [BASS$L_IN_LEN_DT]^2) - 1) DO
297 0674 2 | BEGIN
298 0675 2 | ARRAY_DESC [.COUNTER*%UPVAL, 0, %BPVAL, 0] = .((.ARGLIST [BASS$L_IN_DT_TMT]) + !
299 0676 2 | .DATA_RELOC + (.COUNTER*%UPVAL));
300 0677 2 | END;
301 0678 2 | +
302 0679 2 | Now modify the descriptors. These are usually array descriptors.
303 0680 2 | -
304 0681 2 |
305 0682 2 |
306 0683 2 | INCR COUNTER FROM 0 TO (.ARGLIST [BASS$L_IN_LEN_DM] - 1) DO
307 0684 2 | BEGIN
308 0685 2 | ARRAY_INDEX = .((.ARGLIST [BASS$L_IN_DT_MOD]) + .DATA_RELOC + (.COUNTER*%UPVAL));
```



```
309 0686 3 BSF$A MINOR STG [ARRAY INDEX, 0, %BPVAL, 0] !
310 0687 = .BSF$A_MINOR_STG [ARRAY_INDEX, 0, %BPVAL, 0] + .BSF$A_MINOR_STG;
311 0688 END;
312 0689
313 0690
314 0691 + Allocate dynamic string descriptors.
315 0692 -
316 0693
317 0694 INCR COUNTER FROM 1 TO .ARGLIST [BAS$W_IN_NO_DST] DO
318 0695 BEGIN
319 0696 SP = .SP - %UPVAL;
320 0697 .SP = 0; ! Pointer 0 implies not allocated.
321 0698 SP = .SP - %UPVAL;
322 0699 BLOCK [.SP, DSC$B_CLASS; 0, BYTE] = DSC$K_CLASS_D; ! dynamic
323 0700 BLOCK [.SP, DSC$B_DTYPE; 0, BYTE] = DSC$K_DTYPE_T; ! text
324 0701 BLOCK [.SP, DSC$W_LENGTH; 0, BYTE] = 0; ! length = 0
325 0702 END;
326 0703
327 0704 FMP [BSF$A_STR_DESC] = .SP;
328 0705 +
329 0706 - Allocate fixed string templates.
330 0707
331 0708
332 0709 INCR COUNTER FROM 1 TO .ARGLIST [BAS$W_IN_NO_FST] DO
333 0710 BEGIN
334 0711 SP = .SP - %UPVAL;
335 0712 .SP = 0; ! Pointer 0 implies not allocated.
336 0713 SP = .SP - %UPVAL;
337 0714 BLOCK [.SP, DSC$B_CLASS; 0, BYTE] = DSC$K_CLASS_S; ! fixed
338 0715 BLOCK [.SP, DSC$B_DTYPE; 0, BYTE] = DSC$K_DTYPE_T; ! text
339 0716 BLOCK [.SP, DSC$W_LENGTH; 0, BYTE] = 0; ! length = 0
340 0717 END;
341 0718
342 0719 +
343 0720 - Allocate numeric array elements. They are all initialized to zero.
344 0721
345 0722
346 0723 INCR COUNTER FROM 1 TO (.ARGLIST [BAS$L_IN_LEN_NA]^2) DO
347 0724 BEGIN
348 0725 SP = .SP - %UPVAL;
349 0726 .SP = 0;
350 0727 END;
351 0728
352 0729 +
353 0730 - Allocate temporary cells.
354 0731
355 0732
356 0733 IF ((.ARGLIST [BAS$L_IN_NO_TST] NEQ 0) OR (.ARGLIST [BAS$L_IN_NO_NMT] NEQ 0))
357 0734 THEN
358 0735 BEGIN
359 0736 +
360 0737 - We must set up R9. First allocate string temporaries.
361 0738
362 0739
363 0740 INCR COUNTER FROM 1 TO .ARGLIST [BAS$L_IN_NO_TST] DO
364 0741 BEGIN
365 0742 SP = .SP - %UPVAL;
```



```

366      0743 4      .SP = 0;                                ! Pointer 0 implies not allocated.
367      0744 4      SP = .SP - %UPVAL;
368      0745 4      BLOCK [.SP, DSC$B_CLASS; 0, BYTE] = DSC$K_CLASS_D; ! dynamic
369      0746 4      BLOCK [.SP, DSC$B_DTYPE; 0, BYTE] = DSC$K_DTYPE_T; ! text
370      0747 4      BLOCK [.SP, DSC$W_LENGTH; 0, BYTE] = 0; ! length = 0
371      0748      END;
372      0749
373      0750      !+
374      0751      Point R9 to the last string descriptor allocated.
375      0752      !-
376      0753      BSF$A_TEMP_STG = .SP;
377      0754      !+
378      0755      Now allocate numeric temporaries.
379      0756      !-
380      0757      SP = .SP - .ARGLIST [BAS$L_IN_NO_NMT];
381      0758      END;
382      0759
383      0760      !+
384      0761      Store R9 in the stack frame for setting up I/O lists.
385      0762      !-
386      0763      FMP [BSF$A_BASE_R9] = .BSF$A_TEMP_STG;
387      0764      !+
388      0765      Complete frame.
389      0766      !-
390      0767      FMP [BSF$A_BASE_SP] = .SP;
391      0768      FMP [BSF$A_HANDLER] = BAS$HANDLER;
392      0769      !+
393      0770      First consistency checks.
394      0771      !-
395      0772
396      0773      IF (((.AP [0]) AND 255) NEQ .ARGLIST [BAS$B_IN_NO_FML])
397      0774      THEN
398      0775      !+
399      0776      The number of arguments is incorrect.
400      0777      !-
401      0778      BEGIN
402      0779
403      0780      IF (((.AP [0]) AND 255) GTRU .ARGLIST [BAS$B_IN_NO_FML])
404      0781      THEN
405      0782      BAS$$SIGNAL (BAS$K_TOOMANARG)
406      0783      ELSE
407      0784      BAS$$SIGNAL (BAS$K_TOOFEWARG);
408      0785
409      0786      END;
410      0787
411      0788      IF (((.FMP [BSF$W_FCD_FLAGS]) AND (BSF$M_FCD_RSTR)) NEQ 0)
412      0789      THEN
413      0790      BEGIN
414      0791
415      0792      LOCAL
416      0793      STR_DESC_ADDR : REF BLOCK [8, BYTE];
417      0794
418      0795      !+
419      0796      This procedure has been marked by the compiler as returning a
420      0797      string result. Be sure that there is at least one formal, and
421      0798      that it is a dynamic string descriptor. If so, null its value.
422      0799      !-
```



```

: 423      0800      3
: 424      0801      3
: 425      0802      3
: 426      0803      3
: 427      0804      3
: 428      0805      3
: 429      0806      4
: 430      0807      4
: 431      0808      4
: 432      0809      4
: 433      0810      4
: 434      0811      4
: 435      0812      4
: 436      0813      4
: 437      0814      4
: 438      0815      4
: 439      0816      4
: 440      0817      4
: 441      0818      4
: 442      0819      4
: 443      0820      4
: 444      0821      4
: 445      0822      4
: 446      0823      4
: 447      0824      4
: 448      0825      1

      IF (.ARGLIST [BASS$B_IN_NO_FML] LSSU 1) THEN BASS$$SIGNAL (BASS$K_TOOFEWARG);

      STR_DESC_ADDR = AP [1];
      STR_DESC_ADDR = ..STR_DESC_ADDR;

      IF ((.STR_DESC_ADDR [DSC$B_CLASS] NEQU DSC$K_CLASS_D) OR
          (.STR_DESC_ADDR [DSC$B_DTYPE] NEQU DSC$K_DTYPE_T))
      THEN
          BASS$$SIGNAL (BASS$K_ARGDONMAT);

      !+
      ! Null the string. This insures that, if the procedure does not reference
      ! the string, the function will have the value of the null string.
      !-
      STR$FREE1_DX (.STR_DESC_ADDR);
      END;

      !+
      ! Put the return address back on the stack so we can return to the
      ! caller.
      !-
      SP = .SP - %UPVAL;
      .SP = .RETURN_ADDRESS;
      RETURN;
      END;
```

! of BASS\$INIT_DEF_R8

.TITLE BASS\$INIT_DEF
.IDENT \1-012\.EXTRN BASS\$\$SIGNAL, STR\$FREE1_DX
.EXTRN BASS\$HANDLER, BASS\$K_TOOFEWARG
.EXTRN BASS\$K_TOOMANARG
.EXTRN BASS\$K_SCAFACINT
.EXTRN BASS\$K_PROLOSSOR
.EXTRN BASS\$K_ARGDONMAT
.EXTRN BASS\$K_NOTIMP

.PSECT _BASS\$CODE, NOWRT, SHR, PIC, 2

57	51	D0	00000	BASS\$INIT_DEF_R8::		
54	50	D0	00003	MOVL	R1, R7	0515
52	6E	D0	00006	MOVL	R0, R4	
01	04	A4	91 00009	MOVL	(SP), RETURN_ADDRESS	0591
	0B	13	0000D	CMPB	4(ARGLIST), #1	0596
	08	13	0000D	BEQL	1\$	
7E	00G	8F	9A 0000F	MOVZBL	#BASS\$K_NOTIMP, -(SP)	
00		01	FB 00013	CALLS	#1, BASS\$\$SIGNAL	
53		5D	D0 0001A	MOVL	FP, FMP	0601
5E	D8	A3	9E 0001D	MOVAB	-40(R3), SP	0602
64		3C	E1 00021	BBC	#60, (ARGLIST), 2\$	0612
5E		04	C2 00025	SUBL2	#4, SP	0615
6E		01	D0 00028	MOVL	#1, (SP)	0616
		05	11 0002B	BRB	3\$	0612
5E		04	C2 0002D	SUBL2	#4, SP	0620
		6E	D4 00030	CLRL	(SP)	0621

		5A	81	AE	9E	00032	3\$:	MOVAB	-127(SP), BSFSA_MINOR_STG	0627
			FC	A3	D4	00036		CLRL	-4(FMP)	0631
	FO	A3		5A	7D	00039		MOVQ	BSFSA_MINOR_STG, -16(FMP)	0633
	EE	A3		2C	90	0003D		MOVB	#44, -28(FMP)	0634
	EE	A3	05	A4	90	00041		MOVB	5(ARGLIST), -27(FMP)	0635
	EE	A3	06	A4	B0	00046		MOVW	6(ARGLIST), -26(FMP)	0636
	EB	A3	08	B4	47	9E	0004B	MOVAB	@8(ARGLIST)[DATA_RELOC], -24(FMP)	0637
	D8	A3		54	D0	00051		MOVL	ARGLIST, -40(FMP)	0638
	DC	A3		57	D0	00055		MOVL	DATA_RELOC, -36(FMP)	0639
				50	D4	00059		CLRL	COUNTER	0644
				05	11	0005B		BRB	5\$	
		5E		04	C2	0005D	4\$:	SUBL2	#4, SP	0646
				6E	D4	00060		CLRL	(SP)	0647
F6		50	10	A4	F3	00062	5\$:	AOBLEQ	16(ARGLIST), COUNTER, 4\$	0644
		50	14	A4	9A	00067		MOVZBL	20(ARGLIST), R0	0654
		50		6C	91	0006B		CMPB	(AP), R0	
				03	1E	0006E		BGEQU	6\$	
		50		6C	9A	00070		MOVZBL	(AP), R0	
				50	D6	00073	6\$:	INCL	COUNTER	
				07	11	00075		BRB	8\$	
		5E		04	C2	00077	7\$:	SUBL2	#4, SP	0656
		6E		6C	40	D0	0007A	MOVL	(AP)[COUNTER], (SP)	0657
		F6		50	F5	0007E	8\$:	SOBGR	COUNTER, 7\$	0654
		5E	18	A4	C2	00081		SUBL2	24(ARGLIST), SP	0663
		50		5E	D0	00085		MOVL	SP, ARRAY_DESC	0667
56	18	A4	FE	8F	78	00088		ASHL	#-2, 24(ARGLIST), R6	0673
		51		01	CE	0008E		MNEGL	#1, COUNTER	0675
				0A	11	00091		BRB	10\$	
55		57	1C	A4	C1	00093	9\$:	ADDL3	28(ARGLIST), DATA_RELOC, R5	0676
	60	41		65	41	D0	00098	MOVL	(R5)[COUNTER], (ARRAY_DESC)[COUNTER]	0675
F2		51		56	F2	0009D	10\$:	AOBLSS	R6, COUNTER, 9\$	0673
		51		01	CE	000A1		MNEGL	#1, COUNTER	0685
				0F	11	000A4		BRB	12\$	
55		57	24	A4	C1	000A6	11\$:	ADDL3	36(ARGLIST), DATA_RELOC, R5	
		50		65	41	D0	000AB	MOVL	(R5)[COUNTER], ARRAY_INDEX	
				60	4A	9F	000AF	PUSHAB	(ARRAY_INDEX)[BSFSA_MINOR_STG]	0687
		9E		5A	C0	000B2		ADDL2	BSFSA_MINOR_STG, @ (SP)+	
EC		51	20	A4	F2	000B5	12\$:	AOBLSS	32(ARGLIST), COUNTER, 11\$	0683
		51	28	A4	3C	000BA		MOVZWL	40(ARGLIST), R1	0694
				50	D4	000BE		CLRL	COUNTER	
				0F	11	000C0		BRB	14\$	
		5E		04	C2	000C2	13\$:	SUBL2	#4, SP	0696
				6E	D4	000C5		CLRL	(SP)	0697
		5E		04	C2	000C7		SUBL2	#4, SP	0698
	ED	6E	020E0000	8F	D0	000CA		MOVL	#34471936, (SP)	0701
		50		51	F3	000D1	14\$:	AOBLEQ	R1, COUNTER, 13\$	0694
		A3		5E	D0	000D5		MOVL	SP, -32(FMP)	0704
	EO	51		A4	3C	000D9		MOVZWL	42(ARGLIST), R1	0709
			2A	50	D4	000DD		CLRL	COUNTER	
				0F	11	000DF		BRB	16\$	
		5E		04	C2	000E1	15\$:	SUBL2	#4, SP	0711
				6E	D4	000E4		CLRL	(SP)	0712
		5E		04	C2	000E6		SUBL2	#4, SP	0713
		6E	010E0000	8F	D0	000E9		MOVL	#17694720, (SP)	0716
ED		50		51	F3	000F0	16\$:	AOBLEQ	R1, COUNTER, 15\$	0709
50	2C	A4	FE	8F	78	000F4		ASHL	#-2, 44(ARGLIST), R0	0723
				51	D4	000FA		CLRL	COUNTER	

			05	11	000FC	BRB	18\$		
	5E		04	C2	000FE	SUBL2	#4, SP		0725
			6E	D4	00101	CLRL	(SP)		0726
F7	51		50	F3	00103	AOBLEQ	R0, COUNTER, 17\$		0723
		30	A4	D5	00107	TSTL	48(ARGLIST)		0733
			05	12	0010A	BNEQ	19\$		
		34	A4	D5	0010C	TSTL	52(ARGLIST)		
			1F	13	0010F	BEQL	22\$		
			50	D4	00111	CLRL	COUNTER		0740
			0F	11	00113	BRB	21\$		
	5E		04	C2	00115	SUBL2	#4, SP		0742
			6E	D4	00118	CLRL	(SP)		0743
	5E		04	C2	0011A	SUBL2	#4, SP		0744
	6E	020E0000	8F	D0	0011D	MOVL	#34471936, (SP)		0747
EC	50	30	A4	F3	00124	AOBLEQ	48(ARGLIST), COUNTER, 20\$		0740
	59		5E	D0	00129	MOVL	SP, BSFSA_TEMP_STG		0753
	5E	34	A4	C2	0012C	SUBL2	52(ARGLIST), SP		0757
	A3		59	D0	00130	MOVL	BSFSA_TEMP_STG, -20(FMP)		0763
EC	A3		5E	D0	00134	MOVL	SP, -8(FMP)		0767
F8	63	00000000G	00	9E	00138	MOVAB	BASSHANDLER, (FMP)		0768
	14		6C	91	0013F	CMPB	(AP), 20(ARGLIST)		0773
			13	13	00143	BEQL	25\$		
			06	1B	00145	BLEQU	23\$		0780
	7E	00G	8F	9A	00147	MOVZBL	#BASSK_TOOMANARG, -(SP)		0782
			04	11	0014B	BRB	24\$		
	7E	00G	8F	9A	0014D	MOVZBL	#BASSK_TOOFEWARG, -(SP)		0784
00000000G	00		01	FB	00151	CALLS	#1, BASS\$SIGNAL		
37	E6		0D	E1	00158	BBC	#13, -26(FMP), 29\$		0788
		14	A4	95	0015D	TSTB	20(ARGLIST)		0801
			0B	12	00160	BNEQ	26\$		
	7E	00G	8F	9A	00162	MOVZBL	#BASSK_TOOFEWARG, -(SP)		
00000000G	00		01	FB	00166	CALLS	#1, BASS\$SIGNAL		
	53	04	AC	9E	0016D	MOVAB	4(AP), STR_DESC_ADDR		0803
	53		63	D0	00171	MOVL	(STR_DESC_ADDR), STR_DESC_ADDR		0804
	02	03	A3	91	00174	CMPB	3(STR_DESC_ADDR), #2		0806
			06	12	00178	BNEQ	27\$		
	0E	02	A3	91	0017A	CMPB	2(STR_DESC_ADDR), #14		0807
			0B	13	0017E	BEQL	28\$		
	7E	00G	8F	9A	00180	MOVZBL	#BASSK_ARGDONMAT, -(SP)		0809
00000000G	00		01	FB	00184	CALLS	#1, BASS\$SIGNAL		
			53	DD	0018B	PUSHL	STR_DESC_ADDR		0815
00000000G	00		01	FB	0018D	CALLS	#1, STR\$FREE1_DX		
	5E		04	C2	00194	SUBL2	#4, SP		0822
	6E		52	D0	00197	MOVL	RETURN_ADDRESS, (SP)		0823
			05	0019A	RSB				0825

; Routine Size: 411 bytes, Routine Base: _BASSCODE + 0000

:	449	0826	1	
:	450	0827	1	END
:	451	0828	1	
:	452	0829	0	ELUDOM

BASSINIT_DEF
1-012

F 2
16-Sep-1984 00:35:27
14-Sep-1984 11:55:07

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASINIDEF.B32;1

Page 13
(3)

PSECT SUMMARY

```
:
:      Name      Bytes      Attributes
:  _BASSCODE      411 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)
```

Library Statistics

```
:
:      File      ----- Symbols ----- Pages Processing
:                  Total   Loaded   Percent   Mapped   Time
:  _$255$DUA28:[SYSLIB]STARLET.L32;1      9776         6         0       581      00:01.1
```

COMMAND QUALIFIERS

```
:
:  BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:BASINIDEF/OBJ=OBJ$:BASINIDEF MSRC$:BASINIDEF/UPDATE=(ENH$:BASINIDEF
:  )
```

```
: Size:          411 code + 0 data bytes
: Run Time:       00:12.8
: Elapsed Time:   00:26.6
: Lines/CPU Min:  3901
: Lexemes/CPU-Min: 20541
: Memory Used:    176 pages
: Compilation Complete
```


0024 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

